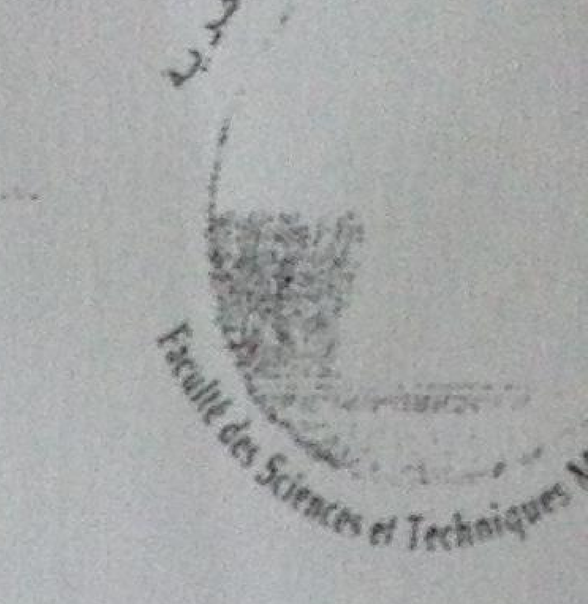
جامعة الحسن الثاني بالدار البيضاء
Université Hassan II de Casablanca
كلية العلوم والتقنيات المحمدية
Faculté des Sciences et Technique- Mohammedia
Département Informatique
F.KHOUKHI



TD 1
MODULE I133

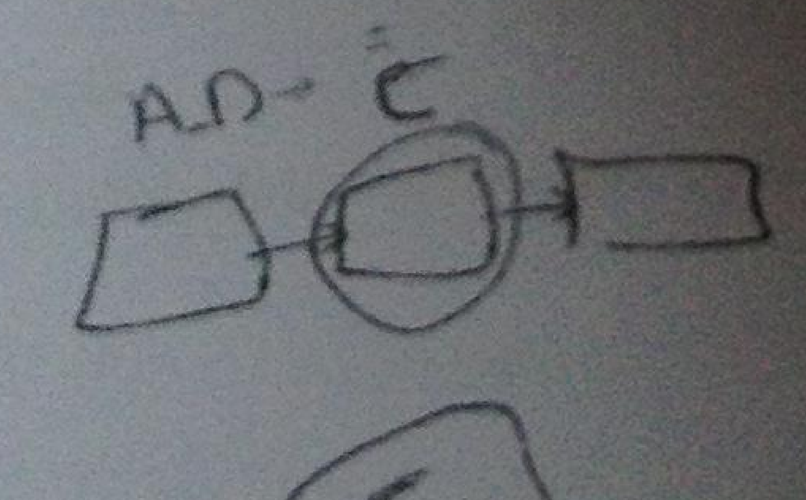
Exercice 1

Traduire cette procédure algorithmique en une fonction en langage C pour créer une liste chainée d'entiers

```
Procedure Creation_liste( Liste *Premier)
Var Donnee : element ;
        Fin :boolean ;
        rep : chaîne de car ;
        Dernier :pointeur sur liste ;
        Avt_Dernier : pointeur sur liste ;
Debut
Lire(Donnee) ;
Dernier=(liste)malloc(sizeof(struct cellule));
Premier=dernier ;
Premier->element=Donnee ;
Fin =Faux ;
Tant que (Non Fin) Faire
Deb
  Avt_Dernier =dernier ;
Ecrire(vous voulez inserer un autre element) ;
Lire(rep) ;
Si rep== "oui" Alors
Deb
Dernier=(liste)malloc(sizeof(struct cellule))
   Avt_dernier->suivant=dernier ;
   Lire(Donnee) ;
   Dernier->element=Donnee ;
Fin
Sinon Fin=Vrai
Fin
Dernier->suivant=NULL ;
Fin
```

2. Ecrire une fonction qui insère un entier saisi en tête de liste
3. Ecrire une fonction qui ajoute un entier saisi en queue de liste
4. Ecrire une fonction qui modifie un entier par un autre
5. Ecrire une fonction qui supprime toutes les cellules qui portent des entiers négatifs
6. Ecrire une fonction qui affiche toutes les cellules de la liste chainée
7. Faire un programme qui propose un menu de choix

Exercice Optionnel







جامعة الحسن الثاني بالدار البيضاء

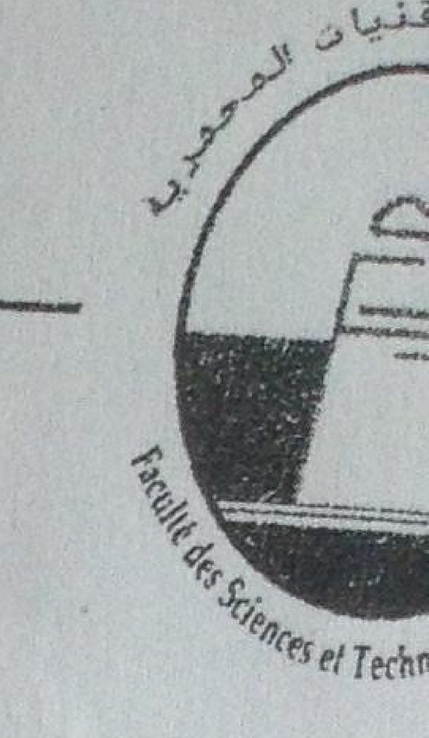
Université Hassan II de Casablanca

كلية العلوم والتقنيات المحمدية

Faculté des Sciences et Technique- Mohammedia

Département Informatique

F.KHOUKHI



TD2 MIP

Exercice 1

On considérera dans une liste chaînée de ce type :

Type Cellule = Structure

Info : chaîne de caractères

Suivant : Liste

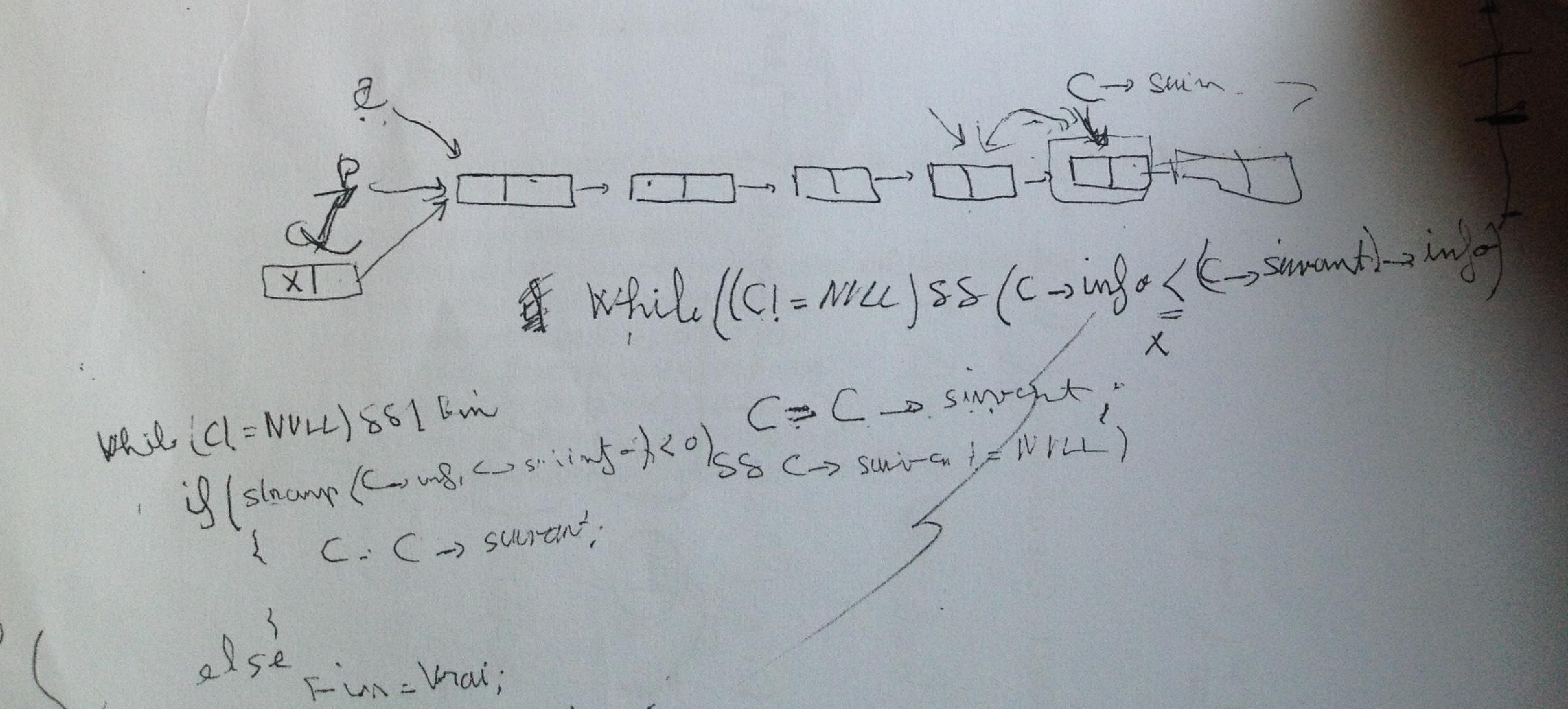
Fin structure

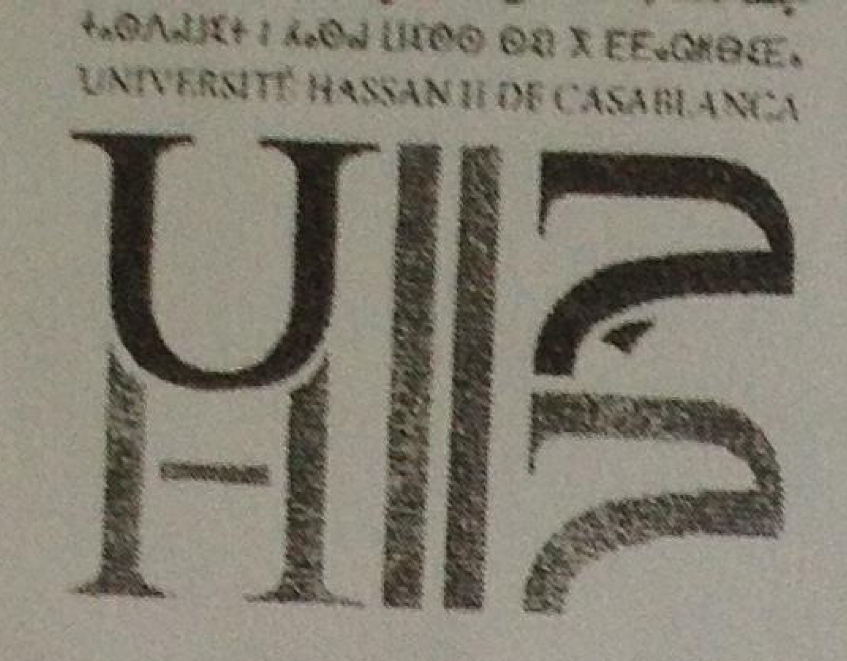
Tete→ Info Suivant → Info Suivant → Info Suivant → NULL

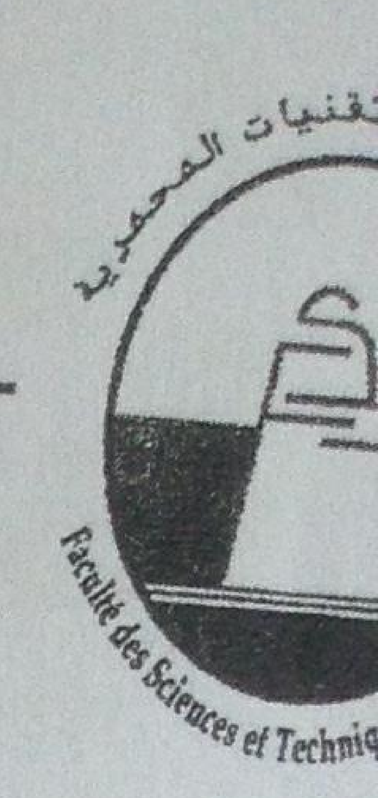
1) Ecrire une fonction qui renvoie le nombre d'éléments d'une liste chaînée.
2) Ecrire une fonction qui renvoie le nombre d'éléments d'une liste chaînée ayant une valeur donnée (champ Info).
3) Ecrire une fonction qui vérifie si une liste chaînée est triée par valeurs croissantes du champ Info
4) Ecrire une fonction qui insère un nouvel élément en tête d'une liste chaînée.
5) Ecrire une fonction qui insère un nouvel élément en queue d'une liste chaînée.
6) Ecrire une fonction qui crée la liste chaînée.
7) Ecrire une fonction qui affiche la liste chaînée.
8) Faire le programme principal qui présente un menu de choix

Exercice 2

Reprendre l'exercice 1 avec une liste doublement chainée







جامعة الحسن الثاني بالدار البيضاء
Université Hassan II de Casablanca
كلية العلوم والتقنيات المحمدية
Faculté des Sciences et Technique- Mohammedia
Département Informatique
F.KHOUKHI

TD3 MIP

Exercice 1

le programme suivant est un exemple d'utilisation d'une pile par un tableau défini ci-après.:

```
#define Max_Pile 100
// type des éléments
 Type;
// type Pile
typedef struct {
int tab[Max_Pile];
 int s;
} Pile;

int main () {
 Pile p = initialise();
 int x;
 p = empiler(p,50);
 p = empiler(p,5);
 p = empiler(p,20);
 p = empiler(p,10);
 printf(" impression de la pile créée " );
 while (est_vide(p)==0){
         x = tete(p);
         printf(" %d" , x);
         depiler(p);
 }/*au fur et à mesure on vide la pile*/
 return 0;
}
```

1) Ecrire la fonction qui initialise une pile
2) Ecrire la fonction qui teste si la pile est vide.
3) Ecrire la fonction qui retourne la valeur du sommet de la pile.
4) Ecrire la fonction qui empile un élément. ans la pile
5) Ecrire la fonction qui dépile une valeur de la pile

Exercice 2

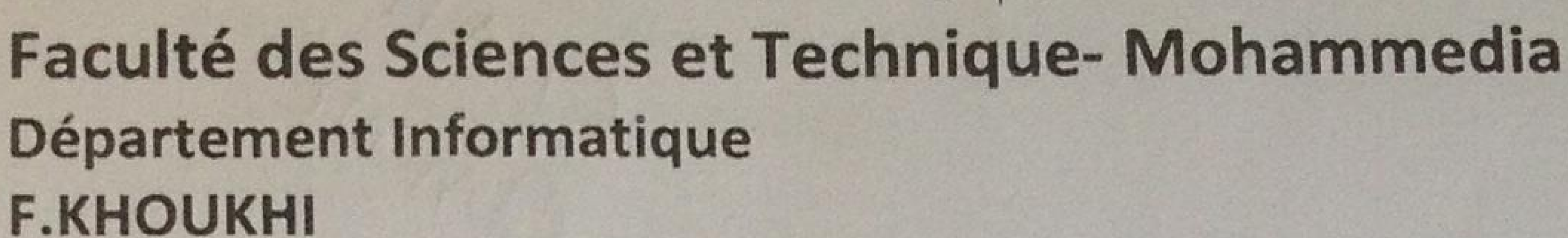
On désire réaliser la notion de pile à l'aide de la structure de donnée définies ci-après.:

```
typedef struct cellule {
        char val ;
        struct cellule *suivant ;
        } Liste ;
```

Tete : pointeur sur le premier élément de la pile. Il doit être nul si la pile est vide.

1) Ecrire une fonction qui initialise une pile
2) Ecrire une fonction qui empile une valeur (champ val).dans une pile
3) Ecrire une fonction qui dépile une valeur de la pile
4) Ecrire une fonction qui teste si la pile est vide.
5) Ecrire une fonction qui teste si la pile est pleine.
6) Ecrire une fonction qui crée la liste chaînée.
7) Ecrire une fonction qui affiche le contenu de la.ple
8) Faire le programme principal qui présente un menu de choix







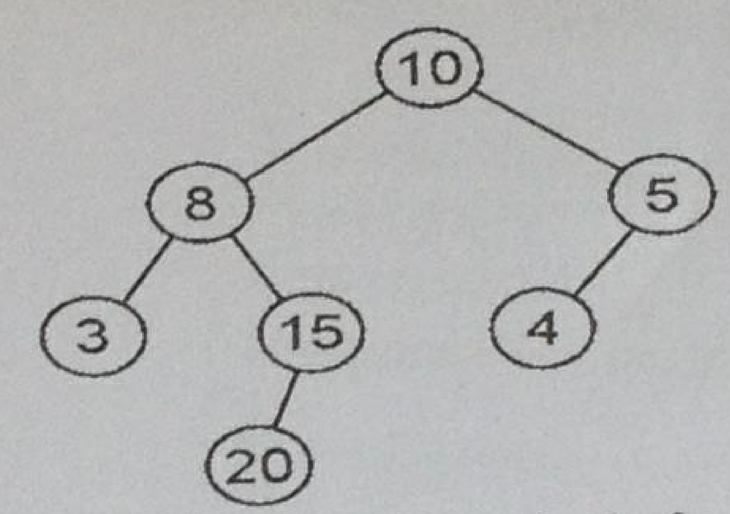
Faculté des Sciences et Technique- Mohammedia
Département Informatique
F.KHOUKHI

TD4 MIP

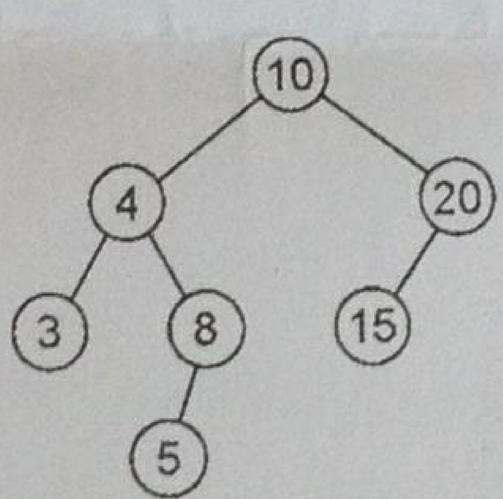
Exercice

Un arbre binaire est un arbre où chaque nœud peut avoir au maximum deux branches. Pour différencier les branches, on les nomme souvent droite ou gauche

```
typedef struct noeud
{ int clef;
struct noeud *gauche, *droite;
} nœud ;
```



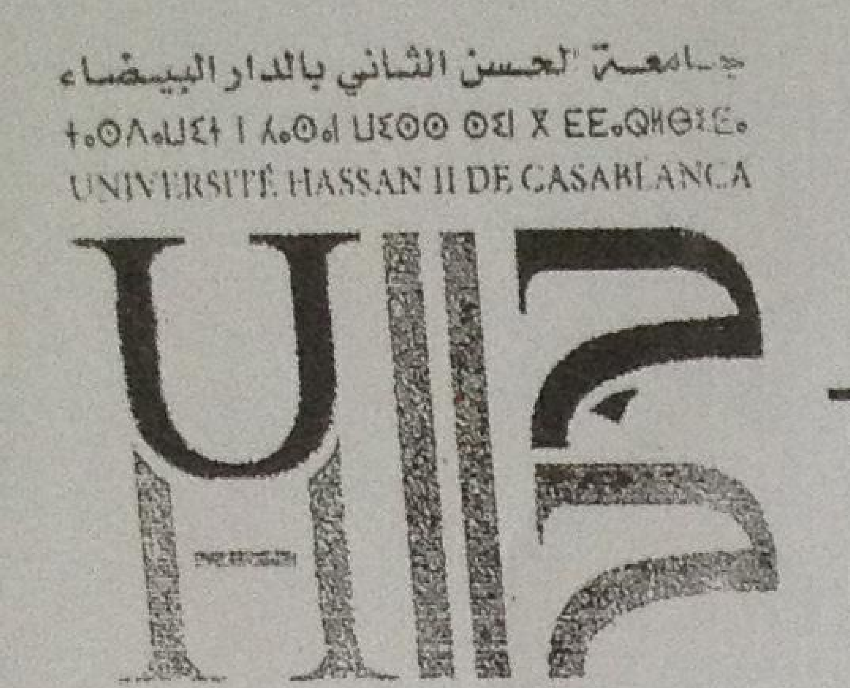
1. Pour qu'un arbre soit efficace, il ne faut pas le remplir anarchiquement, mais de façon ordonnée, ceci afin de retrouver les données rapidement et sans avoir à parcourir l'arbre complet. on trie les éléments à leur insertion dans l'arbre. Le tri sera effectué sur la valeur de la clef. Le premier élément est inséré à la racine de l'arbre, l'élément suivant est inséré à gauche si la valeur de sa clé est inférieure à celle de la racine et à droite si la valeur de sa clé est supérieure à celle de la racine (on aurait pu faire l'inverse). Pour les éléments qui suivent, c'est le même principe jusqu'à trouver un emplacement libre au bout d'une branche. Par exemple, si on avait inséré des éléments ayant comme clé : 10, 20, 4, 8, 5, 15, 3 dans cet ordre, on aurait un arbre équivalent au schéma suivant :




```
int main()
{
  unsigned int clef;
  noeud *Arbre = NULL;
  ajjoutnoeud(&Arbre, 30);
  ajjoutnoeud(&Arbre, 20);
  ajjoutnoeud(&Arbre, 50);
  ajjoutnoeud(&Arbre, 45);
  ajjoutnoeud(&Arbre, 25);
  ajjoutnoeud(&Arbre, 80);
  ajjoutnoeud(&Arbre, 40);
  ajjoutnoeud(&Arbre, 70);
  ajjoutnoeud(&Arbre, 25);
  ajjoutnoeud(&Arbre, 10);
  ajjoutnoeud(&Arbre, 60);
```

```
  puts("------------------------------");
  printTree(Arbre);
  puts("------------------------------");
  Clef = 30;
  if(searchNode(Arbre, Clef)) printf("La cle %d
existe.\n", Clef);
  else printf("La cle %d n'existe pas.\n", Clef);
  Clef = 32;
  if(searchNode(Arbre, Clef)) printf("La cle %d
existe.\n", Clef);
  else printf("La cle %d n'existe pas.\n", Clef);
  puts("------------------------------");
  return 0;
}
```

2. Écrire une fonction Taille(x) prenant un arbre binaire et rendant le nombre de ses éléments.
3. Écrire la fonction ajjoutnoueud qui ajout le nœud dans un arbre ordonné
4. Ecrire la fonction searchnode qui cherche si une clef existe dans l'arbre ou non
5. Ecrire la fonction printTree qui affiche l'arbre en utilisant le parcours ordre (infixé)



جامعة الحسن الثاني بالدار البيضاء
Université Hassan II de Casablanca
كلية العلوم والتقنيات المحمدية
Faculté des Sciences et Technique- Mohammedia
Département Informatique
F.KHOUKHI



TP4 MIP

Exercice 1

Une école propose aux étudiants des formations dans des modules spécifiques. Chaque module comporte un code, un intitulé et nombre d'heures. Les informations concernant les étudiants qui suivent des formations dans des modules spécifiques sont comme suit :

```
struct etd {
char Numero_CNE[12];
char Nom[20];
char Prenom[20]
char intitulé_module[30];
float Moy;
};
```

```
Struct module{
char code[5] ;
Char intitule[30],
Unsigned short int nbr_heure ;
}
```

1. Ecrire une fonction qui ajoute chaque nouveau étudiant inscrit dans une formation dans une liste simplement chainée
2. Ecrire une fonction qui élimine de la liste ci-dessus les maillons correspondant aux étudiants qui ont validé (ayant une moyenne > =10) qui les regroupe dans une nouvelle liste.
3. Ecrire une fonction qui copie la liste simplement chainée des étudiants validés dans un fichier binaire.
4. Ecrire le programme principal

Exercice 2

Un cabinet médical désire informatiser la gestion des rendez-vous de ses patients. Un patient prend RDV soit par Téléphone soit par internet, dans les deux cas, chaque demande de RDV est prise en compte dans une file sous la forme d'une liste simplement chainée. Les me informations concernant les patients sont comme suit :

Struct patient{ char code[5] ; char Nom[10] ;char prenom [10] ; char sexe, date date_RDV, Heure Heure_RDV}

L'application qu'on se propose de réaliser consiste à gérer les informations des patients ensuite afficher à la secrétaire la liste des patients qui doivent consulter le médecin chaque jour.

1. Ecrire une fonction qui crée une file d'attente des patients
2. Ecrire une fonction qui permet à un patient d'annuler son RDV
3. Ecrire une fonction qui permet à un patient de modifier la date et l'heure du RDV
4. Ecrire une fonction qui copie le contenu de la file dans un fichier texte
5. Ecrivez un programme qui permet de réaliser selon le choix de l'utilisateur différentes opérations en utilisant les fonctions ci-dessus.

TD 1 :

Exercice 1 :

① # include <stdio.h>

include <stdlib.h>

include <string.h>

typedef struct liste{ int element;
struct liste * suiv; } liste ;

ici on ne l'a pas fait

↳ si on le met on a plus besoin de la mettre après

enumeration

enum boolean { Faux, vrai } X

liste * Creatliste (liste * Premier) si elle reçoit elle doit faire sortir

{ int Donnee; boolean Fin; char rep [4]; liste * dernier;

liste * Avt_dernier;

Printf ("donnez un entier"); scanf ("%d", &Donnee);

Dernier = (liste *) malloc(sizeof(liste));

Premier = Dernier;

Premier -> element = Donnee;

Fin = Faux;

while (! Fin) Faux ⟺ while (Fin == Faux)

{ Avt_Donner = Dernier;

printf ("voulez vous continuer"); gets (rep);

if (! strcmp (rep, "oui") == 0

{ Dernier = (liste *) malloc (sizeof (liste));

printf ("Donnez un entier"); scanf ("%d", &donnee);

Dernier -> element = Donnee; }

else Fin = vrai;

Dernier -> suivant = NULL;

return premier; }

ajoute en tête :

②
```
liste* insertion (int x, liste *P)
{ liste * D;
  D = (liste *) malloc ( sizeof ( liste));
  D -> suivant = P;
  D -> element = x;
  P = D;     return P; }
```

③ fct qui ajoute en queue :

```
void Ajout ( liste * Premier , int x)
{ liste * courant, dernier;
    courant = premier;
  while (courant -> suivant != NULL)
    courant = courant -> suivant;
  dernier = (liste *) malloc ( sizeof (liste));
    dernier -> element = x;
    courant -> suivant = dernier;
      dernier -> suivant = NULL; }
```

```
liste* dernier(liste * l)
{ liste * p = l;
  if (l)
    { while(p->suivant != NULL)
      p = p->suivant; }
    return (p); }
```

④

premier

[7|] -> [9|] -> [19|] -> [9|X]

p

[x|] -> [9|] -> [19|] -> [9|x]

```
typedef struct cellule { int val;
                struct cellule * suiv; } listeS;

typedef struct cellule { int val;
                struct cellule * suiv;
                struct cellule * prec; } listeD;
```

```
void modifierS( listeS * premier, int E)
{ liste * p; int x;
  p = premier;
    while (p)        = while (p != NULL)
```

6)

TD2 :

1) fct qui renvoie le nombre d'élément :

```
typedef struct cellule { char info [10];
                         struct cellule * suiv; } liste;

int nombreElement ( liste * tete)
 { liste * p = tete;
     int s = 0;
   while ( p! = NULL)
   { s + + ; }
       p = p -> suiv;      return s; }
```

2) fct qui renvoie le nbr d'élément ayant l'info x ;

```
  int nbreElinfo ( liste * tete , char * x)
 { liste * p = tete ; int s = 0;
   while (p)
   { if (strcmp (x, p -> info) = = 0)     s + + ;
          p = p -> suivant ; }
             return s; }
```

3) vérification si une liste est trié

```
             bool Tri ( liste * tete)
   { liste * courant ;
   courant = tete ; bool testord = Vrai ;
    while ( courant ! = NULL)
 { if (strcmp (courant -> info, (courant -> suiv) -> info ) > 0)
    return faux ;
   courant = courant -> suiv ; }
        return (test_ord) }
```

```
default : printf ("choix invalide"); break; }

void affichage (liste* premier)
{ liste* p = premier;
    while (p)
    { puts (p -> info);   p = p -> suiv; }}

main ()
int choix;
char
{ printf ("tapez 1: nbr d'élément (\n");
     a
              2   nbre d'info   \n$.
              b
     tapez    3   Verification
              c
              4   Insertion tête
              d
              5            Queue
              e
              6   Creation
              f
              7   Affichage
              g
     %c        8 Fin
  scanf ("%d", &choix);

  switch (choix)

     { case 'a'
```

TD3 :

exercice 1 :

```
# define MaxPile 100        # define faux 0
  # define vrai 1
typedef struct pile { int tab [Max-Pile];
                           int s;  } Pile;
```

1)
```
void initialise (pile* p)
{   p -> s = -1; }
```

2) /* teste si une pile est ∅ */
```
   int est_vide ( pile* p)
{ if (p -> s == -1)   return vrai;
                      return faux; }
```

3) fct qui retourne la valeur du sommet de la pile:

```
int tete ( pile * p)
{   return p->tab[ p->s] ; }
```

Version 2

```
int tête ( pile p)
{ return p.tab [p. s];}
```

4) fct qui empile une valeur dans la pile.

```
pile Empile (int e, pile p)
{
    if (p.s < Max_Pile)
        { p.s++;    p.tab[ p.S] = e; }
    return p; }
```

5) fct qui dépile un élément.   LIFO

```
pile * depiler ( pile*p)
{   p->s --;
    return p; }
```

Version 2:

```
int depile ( pile * p)
{ int e;
    e = p->tab[ p->sommet]; p->sommet --;
    returne; }
```

main

initialise

Max
3
2
1
0 50

p -> s = -1

pile

empiler (p, 50)

x = tete (p) = 10

1)

2

3)

```
void Enfiler ( char e, FileLC * F)
{ liste * pc;
  pc = (liste*) malloc ( sizeof ( liste));
  if ( pc == NULL)
     { printf ("queue ~~d'ouverture~~"); exit(-1) ;}
     pc -> val = e ;
      pc -> suiv = NULL;
  if ( est vide (F) == vrai)
          { F -> tete = pc ;
              F -> queue = pc; }
  { F -> Queue -> suiv = pc;
       F -> Queue = pc ; }
int tete val ( FileT * F)
   { return F -> tab [F -> tete]; }
Char tete val ( FileLC * F)
    { return F -> tete -> val; }
Void Defiler ( FileLC * F)
 { list * pc ; pc = F -> tete ;
        F -> tete = F -> tete -> suivant ;
             free (pc) ; }
```

```
int recherche noeud (noeud Racine, int e)
{   noeud* A = Racine ;
       while (A! = NULL)

     if (e== A-> Key) return Vrai;
   else if (e> A-> Key) A = A-> droite;
       else  A = A-> gauche;  } return faux;

int taille (noeud * R)
   { if ( R == NULL) return 0;
     else return (1 + taille (R-> gauche + taille (R-> droit)}
   void printArbre (noeud* A)
{ if   (A==NULL)         return ;
     if (A-> gauche) printArbre (A-> gauche)
   printf ( "%d \n" , A-> Key );
   if (A-> droite printArbre(A->droit); }
```